Utilizing AI Planning on the Edge

Ilir Murturi
Distributed Systems Group
TU Wien
Vienna, Austria

Adam Egyed
Distributed Systems Group
TU Wien
Vienna, Austria

Schahram Dustdar
Distributed Systems Group
TU Wien
Vienna, Austria

***Abstract*—The convergence between AI planning techniques and Internet of Things (IoT) can solve various operational and business challenges. However, IoT systems' stringent requirements such as latency and scalability have introduced several challenges to execute and scale planners in cloud environments. Edge computers placed close to the IoT domain (e.g., sensors), can be leveraged for implementing planners and overcome scalability issues. We propose a conceptual framework highlighting executing Expressive Numeric Heuristic Search Planner (ENHSP) on distributed devices in edge networks. As proof of concept, we develop a simulator to show the applicability and feasibility of running planners on the edge. As a case study, we simulate the waste management problem and find the optimal route for disposing waste bins in a city. Throughout the experiments, the user can discover insightful information regarding the planner's applicability on the edge.**

***Index Terms*—Edge-Cloud Continuum, AI Planning, Resource Management**

I. Introduction

The Internet of Things (IoT) has widely disseminated into society, and many services in various industries are built on top of IoT technologies. Managing and controlling a high number of heterogeneous devices (i.e., mobile devices, sensors, etc.) is becoming increasingly complex. On the other hand, the ever-growing number of devices challenges the cloud-centric environments to process incoming sensory data streams within the critical time-frame [1], [2]. In this context, processing IoT data streams over distributed computation entities and gaining authority over a smaller segment of the large network, is critical to achieving various system goals. These distributed computation entities are usually referred to as *edge devices*. Essentially, edge devices allow for pre-processing incoming sensory data streams as well as provides a seamless opportunity to deploy multiple edge applications (i.e., IoT applications) aiming at providing low-latency services for end-users. For instance, a neighborhood may be composed of hundreds of networked devices providing various sensory information about the environment. An edge device in proximity can act as an intermediary computation device to process and store the incoming data. Besides that, an edge device may decide whether the pumped IoT data must be processed at the edge network or forwarded to a cloud infrastructure.

Over the recent years, urban growth became an endemic problem for large cities in the world. In cities with rapid population growth and with high number of visitors, daily public services or public safety services should operate efficiently. In this sense, IoT resources (i.e., sensors, actuators, etc.) distributed over the city, have been seen as critical to improve the quality of life. For example, within a city district, several smart devices provide real-time information about the environment to the closest edge device benefiting from high connectivity to it and its awareness of other resources in its surroundings. An edge device may act as an authority to coordinate available resources in its district and becomes responsible for processing data generated by the smart devices in the field. At the same time, various edge applications (e.g., waste management service) can be deployed on edge, process sensory data, and provide useful information to the end-users. However, not all services can be trivially obtained from IoT resources — for instance, optimizing the energy efficiency of street lighting systems in a neighborhood when no citizen is walking. In that case, several combinations of other IoT devices' resources must be considered to achieve the desired goal [3]. Such a goal can be achieved through utilizing AI planning techniques which are well-known approaches developed to solve planning autonomously and without human intervention. Essentially the planning is defined as the task of coming up with a sequence of actions that will achieve a user goal.

AI planning techniques aim to address NP-hard problems. The bigger the problem instance is, the more time and resources are required to find an optimal solution. In this sense, utilizing the planners in the context of IoT and smart cities becomes an increasingly complex as well as resource-intensive process [4]. Executing AI planning techniques on the cloud is naturally possible. However, in the context of IoT such techniques do not scale very well. In addition, the bandwidth of the networks that carry IoT sensory data streams to and from the cloud hasn't increased appreciably. Essentially, the network bottleneck causes higher latencies than expected response time for IoT systems. Considering the above-mentioned challenges, we argue that Edge Computing can help to (i) tackle the scalability problem by splitting the problem and (ii) overcome networks bottleneck by processing sensory data on the edge. Therefore, this calls for novel resource management approaches, resource coordination, and the need to investigate the feasibility and the applicability of AI planning techniques executed in a distributed manner on the edge. We refer to the applicability aspect by considering real-life scenarios and their concrete requirements to achieve user goals. On the other hand, we refer to the feasibility aspect to show the performance requirements to execute AI planning

models on edge infrastructure.

In this article, we propose a conceptual framework that highlights executing Expressive Numeric Heuristic Search Planner (ENHSP)[1] on distributed edge devices in edge networks. As a proof of concept, we develop a simulator to simulate the waste management problem to find the optimal route for the disposal of waste bins in a city. Subsequently, we discuss the motivation and reasons for choosing to model the waste management problem as well as explain the syntax required to model both the domain and the actual problem for the ENHSP planner. Furthermore, the simulator is configurable and provides functionalities that enable the user to interact at runtime. Throughout the experiments, the user can discover insightful information regarding the planner's applicability on the edge, such as i) performance requirements when executing the planner in different city areas with different parameter settings, and ii) optimizing the overall waste disposal process.

II. Background and Related Work

In the past few years, researchers from academia and industry have been focused on utilizing the power of AI planning techniques in different contexts in edge-based systems (i.e., IoT systems). Several approaches discussed in the literature are based on AI planning techniques such as enabling engineering resource coordination at runtime [3], energy-efficient task offloading [5], deploying self-adaptive IoT systems [6], and adaptation of goal-driven IoT systems [7]. AI-Planning consists of several developed techniques to solve planning and scheduling problems autonomously and without human intervention. Essentially the planning is defined as the task of coming up with a sequence of actions that will lead from an initial state to a specified goal state. A goal can be represented as a single goal state (i.e., predicate) or a complex goals states (i.e., multiple goal states) that can either be true or false. To reach a goal state, the planner is provided with actions that can alter these states. These actions may define preconditions that need to be met to change the states of the problem according to the action's effect. For each task, the planner requires the domain and problem models as inputs (discussed in Section IV-A).

Engineering edge-based systems is challenging, partly due to the heterogeneity, dynamicity, and uncertainty of the devices. Tsigkanos et al. [3] proposed a goal-driven approach for engineering resource coordination at runtime. The approach adopts goal modeling to capture objectives opportunistically at runtime and without any operational status knowledge. Bounded model checking is used as the foundational technique to compute coordination plans that satisfy device goals. Essentially, it considers dependencies among IoT things to achieve a particular goal. Alkhabbas et al. [7] proposed an approach for enabling the automated formation and adaptation of goal-driven IoT systems by exploiting context-awareness and AI-planning techniques. On the contrary to the mentioned works, while focusing primarily on AI planning techniques, we delve more into showing the feasibility and emphasize the applicability of AI planning techniques executed in a distributed manner on the edge. As the case study, we consider an age-old matter and well-known waste management problem. We developed a configurable simulator which is interactable at runtime. The simulator provides functionalities to the user to create diverse use-cases for the ENHSP planner to solve.

[1]The ENHSP Planning System, https://sites.google.com/view/enhsp/

III. Motivation

Providing an efficient waste disposal service in crowded and highly frequented areas (i.e., squares, parks, etc.) by visitors remains a prime concern for metropolitan cities. This is because the cleaner cities are, the more tourists they attract. Parks for instance, may become visitor spots because of their cleanliness, view, and fresh air. On the other side, waste disposal operators traditionally collect waste from bins on a fixed schedule (e.g., hourly or daily). However, the number of visitors and citizens roaming in the various city areas changes frequently. Such public spaces can get crowded very fast, meaning that the waste bin filling rate may increase drastically. In such situations, the fixed schedules become ineffective while driving to an almost empty garbage can causes wasting of resources and money.

To provide real-time scheduling, we assume that several smart vehicles (i.e., smart trucks) may operate continuously around the city areas. Through the integrated IoT sensors, each waste bin provides sensory information to the closest edge device responsible for a particular area in the city (i.e., neighborhood, square, park, or district). On the other side, edge devices are pre-configured regarding the maximum waste level allowed in bins as well as the truck capacity. For instance, when the waste level in bins is above the predefined threshold, a responsible edge device places a lease on the closest available truck to empty all bins in the responsible authority radius. However, we will face an NP-hard problem when considering several factors (i.e., streets, bins, trucks, etc.) to provide most optimal routes for the trucks operating in the city. Therefore, when utilizing AI planning techniques and processing sensory data on the edge, a crucial aspect is to measure performance issues in different scenarios.

IV. Utilizing AI Planning on the Edge

A. Conceptual framework

In the Smart City context, we may have multiple complex problems which can be solved by using planners. More precisely, this means that for each task, we have different planning domains and planning requirements. Thus, this calls for a new framework that allows developers or system administrators to deploy models on available devices at the edge. To overcome the mentioned challenges, we introduce a conceptual framework that enables easy deployment and operation of planners on the edge infrastructure. The conceptual framework is presented in Figure 1.

The proposed framework comprises two main parts: a) a cloud-based IoT platform and b) edge device core functions. The platform consists of three main modules i) Runtime

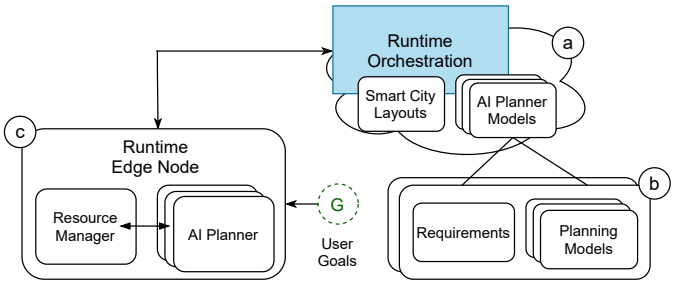

Fig. 1. An overview of the conceptual framework.

Orchestration, ii) Smart City Layouts, and iii) AI Planning Models. The runtime orchestration is responsible for providing an environment where it allows planners' execution in heterogeneous environments. The platform can interact with the edge system components (i.e., resource manager) to access, govern, and orchestrate planners at the edge. For instance, Docker[2] containers are a way to wrap up a planner into its own isolated package. Additionally, the runtime orchestration manages edge devices, automates container provisioning, networking, load-balancing, security, and scaling across the distributed edge devices.

The cloud layer assumes to contain various planning models that a system administrator can upload on distributed edge devices. An edge device may run multiple planners (i.e., ENHSP instances) depending on the device resources. Furthermore, models are written using the Planning Domain Definition Language (PDDL), which is comprised of *planning domain model* and *problem description*. Planning domain model and problem description are essential files required to instantiate a planner (see Section IV-C). In addition to the models, the cloud platform may contain simple city layouts (i.e., maps) or accurate digital architectural models [8].

The second part of the framework represents edge device core functionalities. The Resource Manager is responsible for monitoring the infrastructure-specific metrics such as edge devices' resource capabilities (i.e., hardware). In addition, the module implements an IoT resource discovery mechanism [9] and various communication protocols (i.e., Bluetooth, ZigBee, LoRa, WiFi, etc.) to connect with various surrounding IoT devices. On the other hand, the AI Planning module executes ENHSP instances. The module listens continuously for new requests from the system administrator to start new solver instances. Essentially, after the required files (i.e., domain and problem model) are uploaded, a new container-based service is started at the edge device.

B. Smart City Layout

As outlined in the motivation section, we consider the waste management problem by deploying multiple edge devices and coordinating available resources (i.e., trucks) in the entire city. Before starting with the problem modeling in PDDL, first it is required to relay and document the process of finding the right underlying data-structure for the city itself. Generally speaking, the waste management problem is consolidated to `Nodes` (waste bins), `Streets` connecting them with a given `distance` (representing the cost of the route), and one or multiple `trucks`. However, to visualize the process and enable user interaction with the simulator, we require a static city representation with widespread usage implementation. To overcome such a challenge, we utilize an open-source java library called GeoTools[3] to read, parse, and display the data sets. Thus, the simulator requires the geospatial information in a format called *shapefile*.

C. Use Case: Waste Management Domain Modeling

As outlined in Section IV-A, PDDL is comprised of *planning domain model* and *problem description* files. The domain describes the planning world in a general sense which contains the actions and predicates. Essentially, it defines which objects are existent, the actions that can alter these objects' state, including the pre-conditions that need to be fulfilled, and the effects of these actions on the objects. The domain file doesn't define the set amount of objects (i.e., bins or trucks), neither does it define the planning problem's goal.

Listing 1: Domain-Predicates

```
(:predicates
 (truck ?t - vehicle)
 (bin ?b - wastebin)
 (node ?n - location)
 (street ?s - street)
 (plant ?p - facility)
 (is-at ?p ?n - location)
 (connected ?s - street ?n - location)
 (truck-at ?t - vehicle ?n - location)
 (plant-at ?p - facility ?n- location)
 (emptied ?b - wastebin)
 (empty ?t - vehicle))
```

In Listing 1, we show the exact syntax for defining the domain problem. Most of the predicates are self-explanatory, such that objects are vehicles, bins, streets, etc. In order to derive the city structure from the shapefile retrieved from the OpenGov data, we introduce the notion of whether a node is connected to a street or not. The rest of the predicates express positioning on the map alongside the ones telling the planner whether a waste bin has been emptied or a vehicle is empty after visiting a waste disposal facility and regaining full capacity.

With the `:fluents` requirement we are able to create numeric predicates to keep track of the `capacity` of the *vehicle* or the current `status` of the *waste bins* or assign a numeric cost called `distance` for travelling along a *street* (see Listing 2). The most important function is the `(total-cost)` one

[2]Docker, https://docker.com

[3]GeoTools The Open Source Java GIS Toolkit, https://geotools.org/

Listing 2: Domain-Functions

```
(:functions
      (total-cost)
      (distance ?s - street)
      (capacity ?t - vehicle)
      (wastebin-status ?b - wastebin)
      (wastebin-capacity)
      (max-capacity ?t - vehicle))
```

that is getting modified after every movement of the truck and essentially keeps track of the plan's current cost. This is the metric that will ultimately need to be optimized by the planner. Furthermore, vehicle move actions and preconditions considered in the problem are rather straightforward (i.e., pick up bin, move truck, etc.). Further modeling details can be found in [10]).

D. Problem Skeleton

The problem description file defines the initial and the goal states. Thus, once the format of the domain is set, the problem file is initialized with the actual part of the city that needs to be planned including *nodes*, *streets*, *facilities*, *waste bins* and the *trucks*. To initiate the planner, we require the planning domain and problem as two static files. However, edge devices receive real-time status values from smart devices, and these data need to be translated and inputted into the planner continuously. Thus, to build our simulator and enable the user to interact with it, we generate the static file *on the fly*. This calls for creating a skeleton of the actual problem file, populated by real-time values and edge configuration parameters (see Listing 3).

Listing 3: Problem skeleton

```
(define(problem smart-waste)
      (:domain smart-waste)
      (:objects %1$s )
      (:init %2$s )
      (:goal %3$s )
      (:metric minimize (total-cost))
```

As can be noted in Listing 3, the `%1$s` part is where the actual initialization occurs. Essentially, it gets built as a string in the application itself and uses string replacement to populate the placeholders. Ultimately this file gets written to the file system and the planner. Afterwards, the planner receives its input for the problem definition. Furthermore, to cut down generation times, the static part of the problem is generated as soon as an edge device is placed on the map. Given the edge device's authority radius, it is possible to pre-generate the city structure that needs to be planned to speed up the generation process. This means that only the values that are changing over time (i.e., the bin status and the truck's position) need to be processed before the planning can start.

Another issue that may arise is when the smart truck is out of the edge device's authority radius. An edge device can call the closest available smart truck in the city. Solving such a challenge within the problem skeleton becomes unfeasible in performance terms when the truck is too far away (i.e., the entire city needs to be considered). To overcome such a challenge, we create another problem skeleton intending to move the smart truck within the caller radius (i.e., edge radius) in the most optimal way possible.

V. Simulator Implementation

A. Workflow

To show the applicability and feasibility of executing planners on edge networks, we develop a simulator to simulate the waste management problem [10]. The current version of the prototype is written in Java and provides basic functionalities to enable resource coordination (i.e., between edge devices and trucks) and executing multiple planners on the simulated edge devices. The end-user can interact with the simulator at runtime, configure it, and simulate various scenarios. An overview of the three-step configuration process of the simulator is presented in Figure 2.

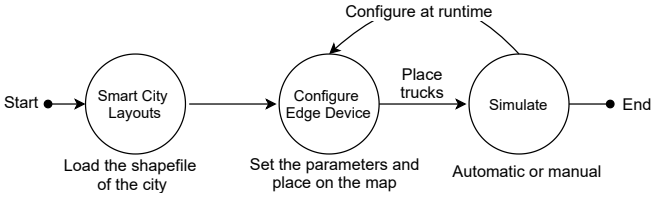

Fig. 2. An overview of the three-step configuration process.

1) Smart City Layouts: We first choose a city that provides data that can be used in conjunction with the ENHSP planner. For instance, the City of Vienna, which is part of the Cooperation Open Government Data Austria[4] provides a wast amount of static and real-time data spanning from general street structure to real-time construction works. The above-mentioned site provides an API for developers to use such data free of charge for their purposes. Nevertheless, since the aim is to develop a stand-alone version of our prototype, downloading the geospatial information in a format called *shapefile* fulfilled the need for offline usage.

2) Configure Edge Device: The end-user can place edge devices anywhere on the map. Prior to its placing on the map, an edge device is essentially configured with the parameters such as *authority radius, number of bins, bin capacity, etc.* The waste bins are distributed randomly inside the edge devices authority zone. For each edge device placed on the map, a new server instance is started on a specific port to generate sensory data for the simulator (i.e., explained in the following subsection). The waste bins positions on the map are generated randomly. Moreover, trucks operate in one of two possible modes: i) `available` and ii) `not-available`.

[4]Cooperation Open Government Data Austria, https://www.data.gv.at/

Fig. 3. An overview of the simulator.

The available mode represents the state when edge devices can place a lease on the truck. The not-available mode represents the state when the truck is leased by another edge device. Besides that, the end-user configures trucks with the following parameters such as *truck capacity, speed, etc.* Each edge device can query at anytime the state of trucks placed on the map.

3) Simulate: The end-user can place trucks at any given place on the map. After the main components are set, the user can either set a fixed interval for plan generation or do it on demand. The plan generation occurs in two steps. First, the edge device places a lease on the truck, and it calculates the fastest route to the closest point in its authority radius. The truck then moves there, and in the second step, the planner generates the most optimal route to empty all the bins in the edge device's authority. Once these bins are emptied, the truck is released, and other edge devices can apply for a lease on the said vehicle. The entire process (i.e., planning and executing the plans) can thus be automated and evaluated over a longer period of time. An overview of the simulator is presented in Figure 3.

B. Simulating sensory data

The time until a waste bin is full usually varies depending on population density and some other variables. For instance, according to the City of Vienna[5], the bins are emptied from once a day to six times a week. To generate waste bins sensory data, we developed a Python-based web server that allows the planner to query actual sensor status and create new instances when the user adds new edge devices on the map.

With generating pseudo-random numbers between 0 and 1 for each bin we get the fill rate function $f(x) = 0.6 + x^3 * (4.167 - 0.6)$. Every random number generated gives us an average of $f(0.5) \approx 1.045\%$ per day. Moreover, waste bins are full until an average of 3.98 days and make it somewhat closer to the situation we encounter in real life (see Figure 4).

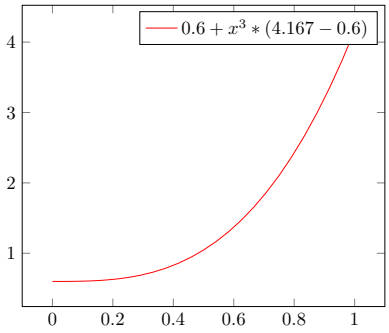

Fig. 4. Fill-rate graph of waste bins

VI. Experiments

Throughout the experiments, the user can discover insightful information regarding the planner's applicability on the edge, such as i) performance requirements when executing the planner in different city areas with different parameter settings and ii) optimizing the waste disposal process. First, we show the planner performance aspects when scaling by the number of trucks considered to generate a single plan. The planner will thus try to find the optimal emptying plan by using one or multiple trucks available. Besides that, we compare the performance scaling by the number of bins considered for generating a single plan. Second, we explain other experiments that the user can realize within the current simulator version.

To monitor resource utilization during the runtime process,

[5]Vienna Waste Disposal Frequency, https://bit.ly/39qpEIN

we use a tool called VisualVM[6] which displays the performance statistics of a specific Java Virtual Machine instance in real-time. Figure 5 presents the planner performance when scaling with increasing the number of trucks assigned to a single edge device. We consider a city area that consists of 129 streets, 8 bins placed within the 71 nodes on the map. As can be noted, with increasing the number of trucks, the average planning time increases as well. The average planning time and memory consumption remain in the acceptable range with three trucks; however, the planner will not scale well when the number of trucks increases. Nevertheless, edge devices are considered resource-constrained devices, and creating multiple edge devices with a single truck responsible for emptying bins at a time overcomes the scalability challenges introduced by the planner.

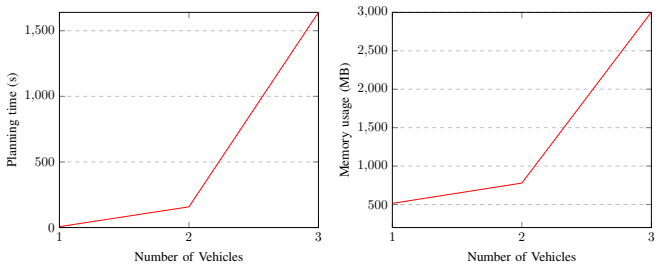

Fig. 5. Scaling by vehicle count.

Figure 6 presents the planner performance when scaling with increasing the number of bins assigned to a single edge device. We consider a similar city area as in the previous experiment. On the contrary to the first experiment, we consider a single truck and the various number of bins. As can be noted, increasing the number of bins in the problem instance results in an average planning time increase. Similarly, as in the first experiment, the planner will not scale well when the number of bins considered in the problem instance is high. Therefore, the user may concretely discover the planner's resource requirements to run it in a particular region of a city.

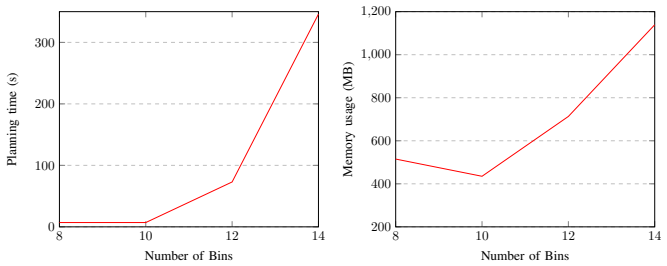

Fig. 6. Scaling by number of bins.

Since we are subdividing the city into small self-governing areas, determining optimal truck count and capacity may determine the overall operating cost. For instance, regularly operating multiple smaller capacity trucks with lower operating costs can be more cost-efficient than having fewer high-capacity trucks covering a larger area. With the parameters revolving around the truck count, we can immediately notice the drop in bins emptied per km driven. Therefore, we can empirically put the cost of dispatching a garbage truck in shorter intervals vs. the cost of doing longer routes but emptying more waste bins along the way.

[6]VisualVM, https://visualvm.github.io/

VII. Conclusion

The developed simulator is the initial attempt at designing a system for executing AI planning techniques in a distributed manner on the edge. The current version is in the early stage of development and provides essential functionalities to simulate the waste management problem. Besides that, edge devices are simulated, and the planner instances for each edge device placed on the map are executed on the same machine. However, when placing multiple edge devices on the map, the simulator may face performance issues. In our future work, we plan to extend the prototype with a module through which developers and users can easily configure edge devices and connect them to their physical entities. Furthermore, we plan to extend the simulator with other AI models supporting various Smart City problems. Finally, we plan to perform a more extensive evaluation of the approach.

Acknowledgment

Research partially supported by the "Smart Communities and Technologies (Smart CT)" and it has received funding from the EU's Horizon 2020 Research and Innovation Programme under grant agreement No. 871525. EU web site for Fog Protect: https://fogprotect.eu/

References

[1] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[2] Schahram Dustdar and Ilir Murturi. Towards distributed edge-based systems. In *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*, pages 1–9. IEEE, 2020.

[3] Christos Tsigkanos, Ilir Murturi, and Schahram Dustdar. Dependable resource coordination on the edge at runtime. *Proceedings of the IEEE*, 107(8):1520–1536, 2019.

[4] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[5] Josip Zilic, Atakan Aral, and Ivona Brandic. Efpo: Energy efficient and failure predictive edge offloading. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 165–175, 2019.

[6] Fahed Alkhabbas, Ilir Murturi, Romina Spalazzese, Paul Davidsson, and Schahram Dustdar. A goal-driven approach for deploying self-adaptive iot systems. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 146–156. IEEE, 2020.

[7] Fahed Alkhabbas, Romina Spalazzese, and Paul Davidsson. Eco-iot: An architectural approach for realizing emergent configurations in the internet of things. In *European Conference on Software Architecture*, pages 86–102. Springer, 2018.

[8] Pedro Martín Lerones, José Llamas Fernández, Álvaro Melero Gil, Jaime Gómez-García-Bermejo, and Eduardo Zalama Casanova. A practical approach to making accurate 3d layouts of interesting cultural heritage sites through digital models. *Journal of Cultural Heritage*, 11(1):1–9, 2010.

[9] Ilir Murturi, Cosmin Avasalcai, Christos Tsigkanos, and Schahram Dustdar. Edge-to-edge resource discovery using metadata replication. In *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–6. IEEE, 2019.

[10] Utilizing AI Planning on the Edge (The Waste Management Problem). https://bitbucket.org/imurturi/utilizing-ai-planning-on-the-edge/.